Zoomable Video Playback on Mobile Devices by Selective Decoding

Feipeng Liu and Wei Tsang Ooi

Department of Computer Science,
National University of Singapore,
13 Computing Drive, Singapore 117417.
liuf0005@gmail.com, ooiwt@comp.nus.edu.sg

Abstract. Modern mobile devices support multi-touch gestures that allow users to naturally zoom into and pan around Web pages, photos, and videos. When users zoom into a video, only part of the region in the video frames are displayed. Ideally, only the regions that the user is viewing are decoded, reducing the computation time (hence increasing the playback frame rate) and power consumption. We call this *selective decoding*. We have implemented a system consisting of an offline analyzer and a mobile video player that implements selective decoding in MPEG-4 Part 2 Simple Profile codec. The analyzer traces various dependency relationships among macroblocks of a given video and produces a meta-data file. The mobile video player supports zoom and pan gestures, and uses the meta-data to trace the macroblocks that are needed to decode the RoI. The player uses a modified decoding process to decode macroblocks selectively based on the trace. Our experiments show that selective decoding can improve playback frame rate by up to 193.3% and reduce energy consumption by up to 64.5%.

Keywords: selective decoding, zoomable video, energy saving, Region of Interest playback

1 Introduction

Due to constraint in the physical form of a mobile device, the display on mobile devices is limited in physical size despite increases in display resolution. To overcome the limited screen size, zoom and pan gestures have been widely adopted to help users to view photos, maps, and Web pages. Recent research indicates zoom and pan are helpful for users when watching videos [1]. When a video is zoomed into, only part of the video is displayed. The visible part is referred to as Region-of-Interest (RoI). A simple approach to display the RoI is to decode the entire frame, crop the RoI, and scale the RoI for display. This approach is inefficient since the entire video frame is decoded but only part of it is needed for display. This observation presents an opportunity to achieve more efficient video playback on mobile devices.

We propose a software approach named *selective decoding* that improves the efficiency of zoomable video playback. As its name suggests, this approach decodes video selectively based on the RoI captured from user interactions. With the improved efficiency, this approach can improve the playback frame rate and reduce the energy consumption.

Our work is based on MPEG4 Part 2 Simple Profile (SP) codec. This approach, however, should be applicable for other Discrete Cosine Transform (DCT) based video codec. We focus our work on the decoder side of the codec in the local playback context, but the principles and techniques could be extended to encoder and video streaming.

The selective decoding approach pre-processes the video to obtain the dependencies among the macroblocks in the video. The dependencies are stored in a meta-data file that is placed along the video. During playback, the video decoder reads the meta-data file along with the video, and computes a bitmask based on the dependencies for every frame on the fly, where the selected macroblocks are marked as '1' and others are marked as '0'. The modified decoder then decodes the selected macroblocks according to the bitmask. In this manner, the macroblocks that are necessary and sufficient to render the RoI are decoded.

The key contributions in our research are as follows. First, we designed a selective decoding process that decodes user specified RoI efficiently. Secondly, we implemented the selective decoding process as a zoomable video player on Android to demonstrate its practical usage. Finally, we experimentally evaluated the frame rate and energy consumption benefits of selective decoding over standard decoding.

The rest of this paper is organized as follows. Section 2 reviews related works. Section 3 analyzes the dependencies in detail and discusses the offline computation of selective decoding. Online computation is covered in Section 4, including the selective mask generation and modified decoder. The implementation of the proposed approach is described in Section 5. We evaluate selective decoding in Section 6 and finally conclude at Section 7.

2 Related Work

Encoding support for zoomable video were explored by several research groups. Mavlankar et al. studied the optimal slice size for zoomable video in a network streaming context [2]. Feng et al. presented how to produce a video stream with RoI cropping support by constraining the video compression process [3].

Other related work exists on zoomable video in the context of video streaming, each with its own focus. Utilizing zoomable video to save bandwidth by refining the encoding process is studied by Ngo et al.[4]; Zoomable video on peer-to-peer streaming is explored by Mavlankar et al.[5]; RoI prediction and tracking for streaming zoomable video has also been examined [6–8]; multiple RoIs support is investigated by Bae et al.[9].

A recent work that is close to ours is the work by Liu et al. [10]. They proposed a partial decoding scheme for H.264 decoder, decoding I-frames fully

and P-frames partially. In partial decoding, they decode all macroblocks within a fixed distance away from the RoI. Their approach does not ensure that all RoI macroblocks are decoded correctly because RoI macroblocks can depend on a macroblock outside the partial decoding area. In contrast, our approach analyze the motion vectors so that all macroblocks are decoded correct and only relevant macroblocks are decoded.

3 Offline Computation

In this section, we elaborate on the pre-processing stage of our selective decoding approach. In this stage, we analyze the dependencies among the macroblocks in the video and store some of the dependency information in a meta-data file to reduce the dependency during decoding. Since it is essential to understand various dependencies in order to comprehend offline computation, we first explain the types of dependencies that exist in MPEG4 SP coding. For all subsequent discussions in this paper, we assume the video is in YCbCr420 color space, which means a macroblock contains four luminance blocks and two chrominance blocks.

3.1 Dependencies

Two categories of dependencies can be identified, namely intra-frame dependency and inter-frame dependency. Dependencies are the reason why some macroblocks outside of RoI need to be decoded. By saving certain information, we can reduce the dependencies and improve the efficiency of selective decoding. Below we analyze the dependencies and introduce the methods to reduce the dependencies..

Intra-frame Dependency Intra-frame dependency refers to the dependencies among macroblocks within a single frame. There are two sources of intra-frame dependency, including DC&AC Prediction and MV coding.

DC&AC Prediction is performed for I-macroblock when the header field short_video_header is set to '0'. It consists of two steps, namely reference block selection and prediction decoding. The reference block selection step can be illustrated by Fig. 1.

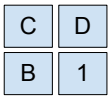


Fig. 1. DC&AC Prediction Reference Block Selection

There are two candidate reference blocks for each block, the immediate left block and the immediate upper block. In addition, the upper left block is also

needed in order to determine the reference block. To determine the reference block for block 1 in Fig. 1, the following rule is applied,

```
if (|F(B)[0][0] - F(C)[0][0]| < |F(C)[0][0] - F(D)[0][0]|)
   predict from block D
else
   predict from block B
```

F(B)[0][0], F(C)[0][0] and F(D)[0][0] refer to inverse quantized DC value of block B, C and D respectively. This rule implies that a block is dependent on three neighboring blocks during the reference block selection process. In prediction decoding, however, the block depends only on the selected reference block.

To reduce the dependencies during the reference block selection process, we store a single bit to indicate the reference block, with '0' indicating up and '1' referring to left. Storing this meta-data reduces the dependency for a block from three to one during decoding.

Since MV of P-frame is differentially coded, it serves as another source of intra-frame dependency. At motion decoding, the decoder recovers the MV values based on the decoded base values and residue values obtained from neighboring macroblocks. In our selective decoding implementation, we store the MVs to trace the inter-frame dependency, which is discussed next. Therefore the MVs can be read directly by decoder and no MV prediction decoding is needed. Thus the dependency due to MV prediction decoding is eliminated completely.

Inter-frame Dependency Inter-frame dependency refers to the dependencies among macroblocks at different frames, which is caused by motion compensation coding. In MPEG4 SP, motion compensation decoding only occurs at P-macroblock of P-frame. The inter-frame dependency is illustrated as Fig. 2.

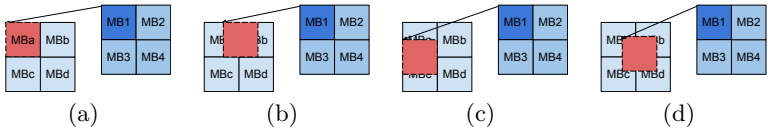


Fig. 2. Different Cases of Motion Compensation Decoding

Motion compensation decoding is performed on MB1 of the current frame, with reference to a region of a previous frame. The number of dependent macroblocks depends on whether the reference region aligns with the macroblock boundary. As shown in Fig. 2, MB1 depends on one macroblock at (a), two macroblocks at (b) and (c), and four macroblocks at (d).

Khiem etc. proposed an approach to reduce dependency due to motion compensation [4]. We adopted their technique in this research. Using the case in Fig. 2(d) as an example, the approach is illustrated Fig. 3.

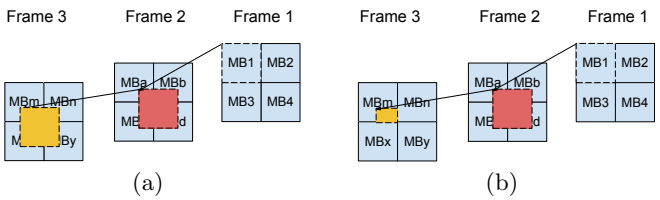


Fig. 3. Dependency Analysis for Motion Compensation

MB1 at Frame 1 depends on four macroblocks at Frame 2. MBa at Frame 2 depends on another four macroblocks at Frame 3. Dependency can be reduced by tracing it at pixel level. The dependency between Frame 1 and Frame 2 remains the same. But we do not really need all pixels at Frame 2 MBa. The region needed at MBa depends on only part of MBm at Frame 3. Therefore, the dependency for MBa between Frame 2 and 3 are reduced from four to one in this example.

3.2 Dependency Files

The offline computation partially decodes a video and records down the dependency information into a set of meta-data files named *dependency files*. The dependency files are generated for each Group of VOP (GOP). For every GOP, the dependency files include the following,

1. GOP record file: it contains the start and end frame numbers of a GOP.
2. MB start and end position file: it stores the macroblock start and end bit positions in the video bitstream for every MB of all frames in a GOP. The positions are needed for the decoder to seek the bits for a MB.
3. DC&AC prediction direction file: it contains the DC&AC prediction direction. A single bit is used to store the direction for each block. The direction is read directly by the decoder to avoid decoding the macroblocks used in DC&AC prediction reference selection but not in actual prediction decoding. The direction is also used to trace the intra frame dependency.
4. MV file: it records the MV values for every macroblock of each P-frame in the GOP and the number of bits for MVs. The selective decoder reads the MV from this file and skip the encoded bits. This file not only eliminates the MV decoding dependency, but also allows the online computation to trace the inter-frame dependency.

We modified the standard MPEG4 SP decoder to partially decode a video in order to generate the above files.

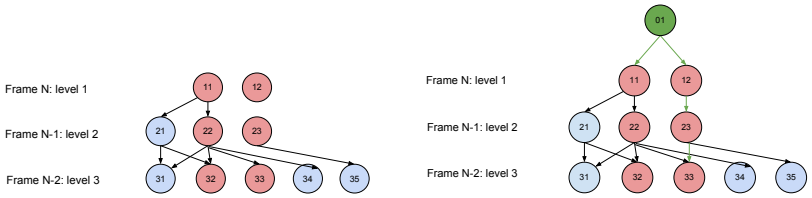
4 Online Computation

Offline computation is done once and the dependency files are saved. Every time the video is played, the selective decoder loads the dependency files, computes a selective mask for each frame, and decodes according to the mask.

4.1 Selective Mask Computation

Selective mask indicates the macroblocks that the decoder needs to decode as '1' and the rest as '0'. It considers both inter-frame and intra-frame dependencies. Note that the inter-frame dependency has to be computed first. If intra-frame dependency is computed first, when computing inter-frame dependency, the computation will select some new P-macroblocks and I-macroblocks. Not computing the intra-frame dependency for the newly selected I-macroblocks would lead to decoding errors at these I-macroblocks. The error will subsequently affect the motion compensation decoding at other macroblocks using these I-macroblocks as reference. In contrast, if inter-frame dependency is computed first, the intra-frame computation will select only I-macroblocks because DC&AC prediction coding only applies to I-macroblock. Since inter-frame dependency does not apply to I-macroblocks, the newly selected I-macroblocks will not introduce errors.

Inter-frame Dependency Computation Inter-frame dependency is caused by motion compensation decoding. A MPEG4 SP GOP consists of an I frame followed by a sequence of P frames. The P-macroblocks of every P frame are motion compensated with reference to macroblocks of its previous frame. Thus, every P frame is dependent on its previous frame. Therefore we compute the inter-frame dependency from last frame back to the first frame of the GOP. The dependencies are shown as Fig. 4(a).

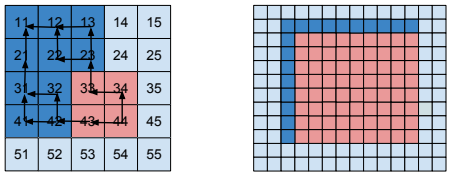


(a) Inter-frame Graph (b) Modified Inter-frame Graph

Fig. 4. Inter-frame Dependency Abstraction

Suppose frame N is the last frame of the GOP, the figure shows the dependencies in three frames. The macroblocks and the dependencies form a graph. By adding a pseudo root node and edges connecting the RoI macroblocks, the graph is transformed to a weakly connected directed acyclic graph shown as Fig. 4(b). With this modification, the graph traversal algorithm Depth-First Traversal (DFT) or Breadth-First Traversal (BFT) can be applied. The macroblocks that are visited by the graph traversal are selected, while the rest are not needed.

Intra-frame Dependency Computation Similar to Inter-frame dependency, intra-frame dependency computation is abstracted as a graph traversal problem.



(a) Graph (b) Optimization

Fig. 5. Intra-frame Dependency and the Optimization

In Fig. 5(a), macroblock 33, 34, 43 and 44 are RoI macroblocks. The dependency graph due to DC&AC prediction coding is depicted. Because the dependency direction is always pointing up or left, the graph rooted at a macroblock is always formed by one of the graphs rooted at the first row and first column macroblocks of the RoI and some macroblocks within the RoI. Therefore, an optimization is to apply graph traversal algorithms only on the macroblocks at upper and left edges of the RoI, and select all macroblocks within RoI. This is illustrated as Fig. 5(b).

4.2 Select the Bits

In order to decode selectively, a mechanism is needed for the decoder to select the bits for selected macroblocks. Two approaches can be used to select the bits, namely bitstream reconstruction and bit seeking. In bitstream reconstruction, a new video bitstream is constructed according to the selective masks and the macroblock start and end positions. The newly constructed bitstream consists of only the macroblocks selected in selective masks. At bit seeking approach, we instruct the decoder to seek to the start position of next selected macroblock at decoding. The second approach avoids the additional memory allocation for the new bitstream therefore it is the preferred approach in our work.

5 Implementation

We implemented the techniques and processes described in previous sections on Android platform as a zoomable video player. The architecture of the player is depicted in Fig. 6.

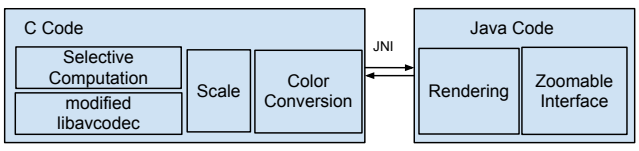


Fig. 6. Zoomable Video Player Implementation

We modified the open source MPEG4 SP codec from libavcodec of ffmpeg 0.7 [11] and added the selective decoding functions. At the time of implementation, the ffmpeg codec was not optimized to run on Android platform, especially the scale and color conversion process. We adopted the code from another two open source projects: scaling from libyuv [12] and color conversion from Google Chromimum project [13].

We implemented the rendering and zoomable interface in Java with Android SDK. The gesture detector detects a user's pinch or pan gesture, and translates the gesture detected as zoom scale and pan scale. We then compute a RoI from these scales based on what user can see on the phone screen.

6 Evaluation

The purpose of selective decoding is to achieve more efficient zoomable video playback. We evaluate the efficiency of selective decoding from two aspects, video playback frame rate and energy consumption.

We used a Samsung Galaxy S2 phone for experiment. The device has a dual core processor of 1200 MHz clock rate each and 1024 MB RAM; it runs Android 2.3.6. Two videos of 1080p are used for evaluation. The first one is a five minute recording of a university lecture. The lecturer is the only person moving around, so there is little motion in the video. The second video captures a scene of a few toys rotating constantly, therefore contains a large amount of motion. The video is about 15 seconds. We refer the lecture recording video as video A and the toy rotation video as video B.

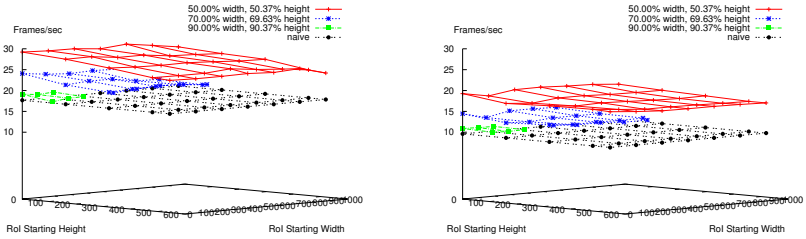
6.1 Frame Rate

Selective video playback frame rate is affected by both RoI size and position. We designed experiments to examine the influence of each.

Different RoI Positions Different regions of a video frame may contain different amount of motions and dependencies, therefore the RoI position can affect the amount of processing at decoding and subsequently the video playback frame rate. We fix the RoI size, and then move the RoI from the upper left corner to the lower right of the video frame. The tests are repeated for three different RoI sizes, with the RoI width and height set as 50%, 70%, and 90% of the original video's width and height.

In Fig. 7, each 3D surface indicates the frame rate for a RoI size. The intersection in a surface indicates the start position of a particular RoI size.

Looking at each 3D surface, the frame rate tends to decrease when the RoI starting width and/or starting height increases. The frame rate decreases because the intra-frame dependencies increase towards the lower right, and more dependencies cause more macroblocks to be selected and decoded. There are, however, exceptions to this decrease trend, which are probably caused by different amount of motions at different RoI positions. Compare different 3D interfaces

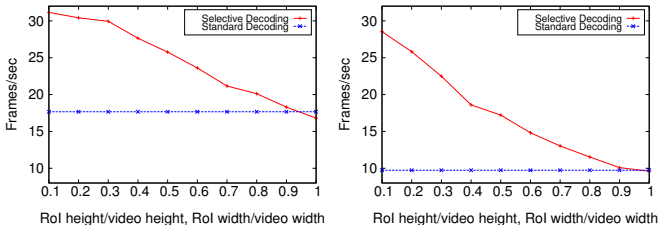


(a) Video A (b) Video B

Fig. 7. Frame Rate with 90%, 70%, and 50% of RoI at Different Positions

at a single figure, it is clear that the RoI size has a more significant effect than RoI position and selective decoding outperforms standard decoding. Compare Fig. 7(a) with (b), the frame rate for video A is higher than video B. This result is expected because video A has less amount of motion than video B.

Different RoI Size Previous experiment already reveals that RoI size has significant influence on video playback frame rate. This experiment examines the affection of RoI size further. We position the RoI at the center of the video frame and vary the RoI height and width from 10% to 100% of original video height and width.



(a) Video A (b) Video B

Fig. 8. Frame Rate with 10% to 100% RoI Centered

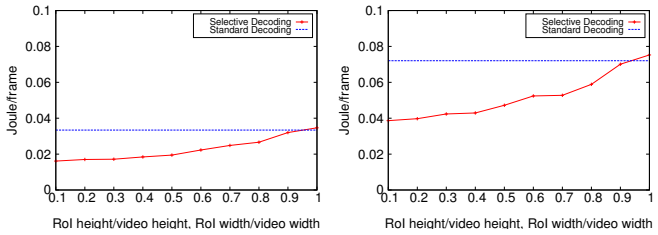
Looking at either Fig. 8(a) or (b), frame rate decreases as RoI size increases. Selective decoding achieves higher frame rate at RoI size smaller than 90%. At 10% RoI, the frame rate is improved by 76.3% and 193.3% for video A and B respectively. The curves at different figures differ due to different amount of motion at center of the video frame.

6.2 Energy Consumption

Energy consumption is the other important aspect of our evaluation. Two experiments are done with different focuses.

PowerTutor Measurements Battery energy is mainly consumed by display and CPU at video playback. PowerTutor [14], an Android power measurement tool, is capable of measuring power consumed by different hardware components for a user specified app. In this experiment, we vary RoI size from 10% to 100%. The frame rate is controlled so that both selective decoding and standard decoding always play at same rate. This control is essential for a fair comparison because the power consumption is dependent on display time.

For display measurements, we found selective decoding and standard decoding consume almost same amount of energy. However, this is not the case for CPU power consumption, which is shown as Fig. 9.



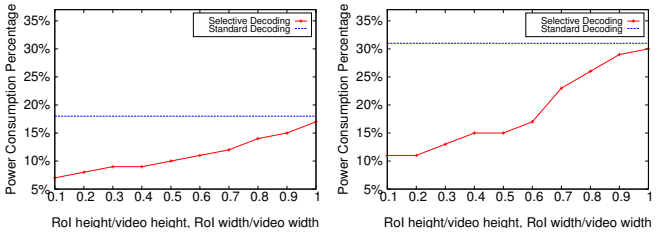
(a) Video A (b) Video B

Fig. 9. CPU Power Consumption Per Frame

Looking at each figure individually, CPU power consumption per frame at selective decoding increases when RoI size increases. At RoI size 90% or less, selective decoding consumes less CPU energy than standard decoding. Compare Fig. 9(a) with (b), video playback for heavy motion video tends to consume more power because of more motion compensation decoding.

Power Drain Experiment PowerTutor measurements show selective decoding can save energy, mainly by reducing CPU power consumption. However, PowerTutor measurement is obtained through offline-training models [14] and may not be accurate for all phones. We designed another experiment to measure the power consumption.

In this experiment, we place RoI at the video frame center and play the video repeatedly with constant frame rate. Based on the processing power and video, we set the frame rate for two videos as 15 FPS and 8FPS respectively. The percentage of power drained is recorded for comparison. Before each test, we fully charge the phone battery to 100% and disable all background activities including Wi-Fi, Bluetooth, GPS, etc.



(a) Video A (b) Video B

Fig. 10. Percentage of Power Drained

Looking at each figure individually, selective decoding consumes less power. At 10% RoI size, the battery consumption is reduced by 61.1% and 64.5%.

In summary, selective decoding proves to be efficient in terms of playback frame rate and battery energy consumption when RoI size is below certain threshold.

7 Conclusion and Future Work

We designed a software approach named selective decoding to reduce the battery power consumption and increase the frame rate for zoomable video playback on mobile devices. Selective decoding is based on analyzing and tracing various intra- and inter-frame dependencies among macroblocks. By doing so, we compute a selective mask which indicates the macroblocks needed to present a clear scene in a user requested RoI and the modified decoder can then decode selectively according to the mask.

Selective decoding illustrates the idea of achieving more efficient zoomable video playback by tracing the dependencies among macroblocks. There are many possible future work can be done. Firstly, the dependency file storage overhead is large. The size of dependency files for video A is about five times the original video size, while the size for video B is 96% of original video size. Because the two videos have same resolution but different bit rates, and dependency file size is dependent on video resolution rather than bit rate, the difference of dependency file size overhead is obvious. We store the dependency information as plain values in binary format. Better storage scheme and compression can be applied to reduce storage overhead. Secondly, the dependency file generation could be done online, which requires optimizing the process and integrating it with selective mask computation. Thirdly, expanding selective decoding to encoder may bring some benefits. The encoder can generate dependency files and control the amount of dependencies among macroblocks. Lastly, selective decoding at decoder may help to improve the existing research on zoomable in network streaming context, which deals with encoders mostly.

Acknowledgement

This research is conducted under the NExT Search Center, supported by the Singapore National Research Foundation and the Interactive Digital Media R&D Program Office of Media Development Authority under research grant WBS:R-252-300-001-490.

References

1. Khiem, N.Q.M., Ravindra, G., Ooi, W.T.: Towards understanding user tolerance to network latency in zoomable video streaming. In: Proceedings of the 19th ACM international conference on Multimedia. MM '11, New York, NY, USA, ACM (2011) 977–980
2. Mavlankar, A., Baccichet, P., Varodayan, D., Girod, B.: Optimal slice size for streaming regions of high resolution video with virtual pan/tilt/zoom functionality. In: Proc. of 15th European Signal Processing Conference (EUSIPCO. (2007)
3. Feng, W.C., Dang, T., Kassebaum, J., Bauman, T.: Supporting region-of-interest cropping through constrained compression. ACM Trans. Multimedia Comput. Commun. Appl. **7**(3) (September 2011) 17:1–17:16
4. Ngo, K.Q.M., Guntur, R., Ooi, W.T.: Adaptive encoding of zoomable video streams based on user access pattern. In: Proceedings of the second annual ACM conference on Multimedia systems. MMSys '11, New York, NY, USA, ACM (2011) 211–222
5. Mavlankar, A., Noh, J., Baccichet, P., Girod, B.: Peer-to-peer multicast live video streaming with interactive virtual pan/tilt/zoom functionality. In: in Proc. of IEEE International Conference on Image Processing (ICIP
6. Mavlankar, A., Varodayan, D., Girod, B.: Region-of-interest prediction for interactively streaming regions of high resolution video. In: In Proc. International Packet Video Workshop. (2007)
7. Shimoga, K.B.: Region-of-interest based video image transcoding for heterogenous client displays. In: Packet Video 2002. (2002)
8. Fan, X., Xie, X., qin Zhou, H., ying Ma, W.: Looking into video frames on small displays. In: In Proc. of ACM Multimedia 2003, Press (2003) 247–250
9. Bae, T.M., Thang, T.C., Kim, D.Y., Ro, Y.M., Kang, J.W., Kim, J.G.: Multiple region-of-interest support in scalable video coding. ETRI Journal (2006) 239–242
10. Liu, C., Jin, X., Zhang, T., Goto, S.: Partial decoding scheme for H.264/AVC decoder. In: Intelligent Signal Processing and Communication Systems (ISPACS), 2010 International Symposium on. (dec. 2010) 1 –4
11. ffmpeg: Ffmpeg. http://ffmpeg.org/index.html (05 2012)
12. libyuv: libyuv. http://code.google.com/p/libyuv/ (5 2012)
13. chromium: chromium. http://code.google.com/p/chromium/issues/detail?id=71403 (5 2012)
14. Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R.P., Mao, Z.M., Yang, L.: Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. CODES/ISSS '10, New York, NY, USA, ACM (2010) 105–114